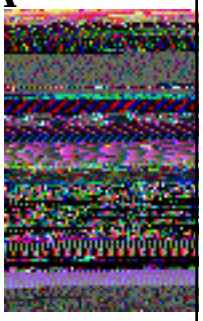
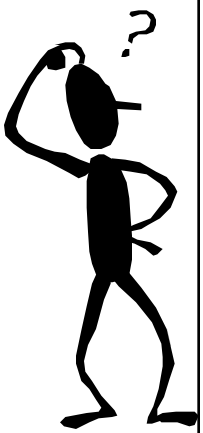
The Alternatives and Supplements to Conventional Testing

- **Quality Week 1998, San Francisco CA**
  - **Www.soft.com**
  - **Wed. 27th May 1998**
  - **Session 4Q 3:30-5:00 PM**
- **Tom Gilb**
  - **Homepage: www.Result-Planning.com**
  - **Iver Holtersvei 2, N-1410 Kolbotn, Norway**
  - **Gilb@ACM.org, +47 66801697**



Purposes of *'Conventional Testing'*

- Evaluate whether Software can be *released*
  - Economically
  - Safely
- *Clean up* software
  - Remove bugs
  - Regression test



First premise of this talk: Test Alternatives

- Any method which can serve the purposes of conventional testing
  - More effectively or
  - Less costly
    - In terms of money, people, time
  - is worthy of being used instead of conventional testing

Talk Premise 2

- Any method which can improve conventional testing process in effect or costs deserves to be considered as a supplement to them.

An alternative to Conventional Testing

- **Would make some type, form, or extent of testing unnecessary**
- **Because the alternative would be more effective and/or less costly**

Some *alternatives*

- **A1. Ship 'testing' to users**
- **A2. Distinct Software**
- **A3. Evolutionary Testing**
- **A4. Defect Detection Inspections**
- **A5. Defect Prevention Process**

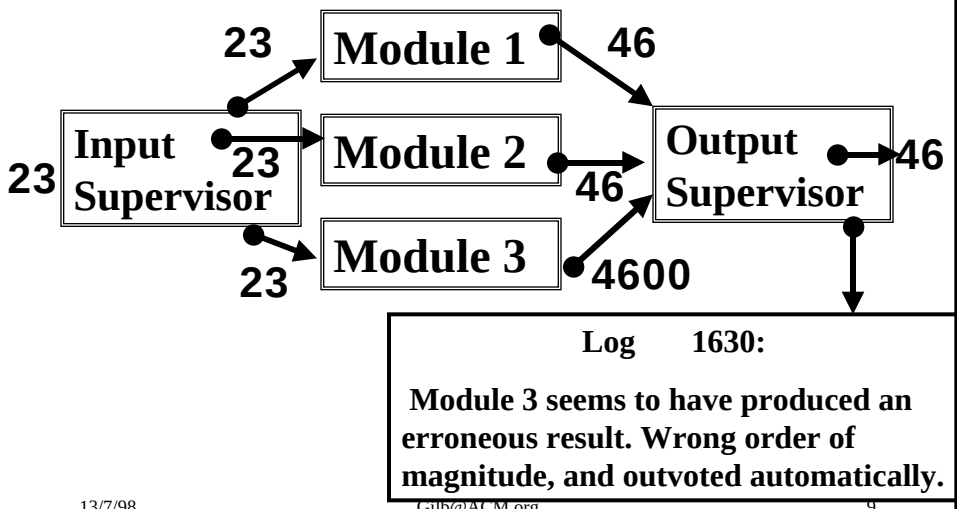
A1. Ship testing to users

- Leon Osterweil U of . Mass. Amhurst at Quality Week 96 "Perpetually Testing Software" 22 May 96.
- Allow users to test software
- with their HW/SW configuations
- Requires much more design for an 'exportable test process'
- Solves the problem of all the possible combinations of equipment and software components now and later, a user can have.
- Gives supplier much better documentation of bug situations

A2. Distinct Software

- 2 or more independent modules
- Should have same outputs for same inputs
- A 'supervisor' checks outputs and reacts
- Reactions are:
  - Log situation
  - Stop the process for manual evaluation
  - Make a choice based on
    - Probably right answer
- Practised at least since 1970s, Space Shuttle etc.
- Also known as 'N-version programming'
- Essentially same principle as a Tandem computer, except in software.
- Allows 'testing' in real time using only real data
- Example: Avizienis 1984 Design Diversity:an approach to fault tolerance of design faults, Proc. NCC AFIPS Press 53, pp163-171.

A2. Distinct Software



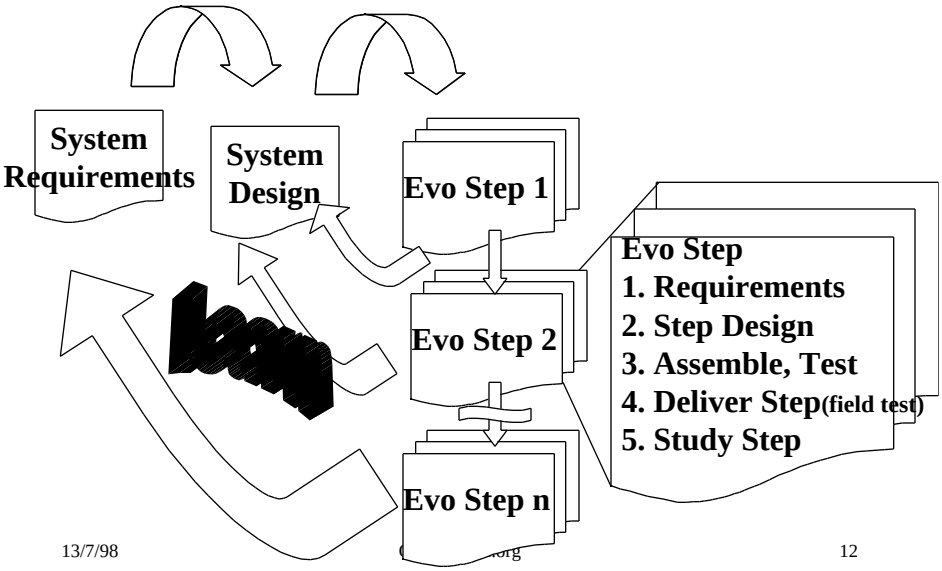
A2. Distinct Software

- The "expected output" is generated by computer, not a human tester.
- Can be used in test mode or operational mode
- Can be used in sampling mode operationally
- Can be used for automatic correction
- Gives differential diagnosis automatically
- Cost of generating extra code is less than the saving in manual test effort (5% coding).
- Distinct modules can be simplified models
  - Example Rockwell's versions of Space Shuttle re-entry versus IBM's.
- Can be used at 'design' level and/or 'coding' level.
  - Example Ramamoorthy UCB Nuclear Plant system 1978.

A3. Evolutionary Testing

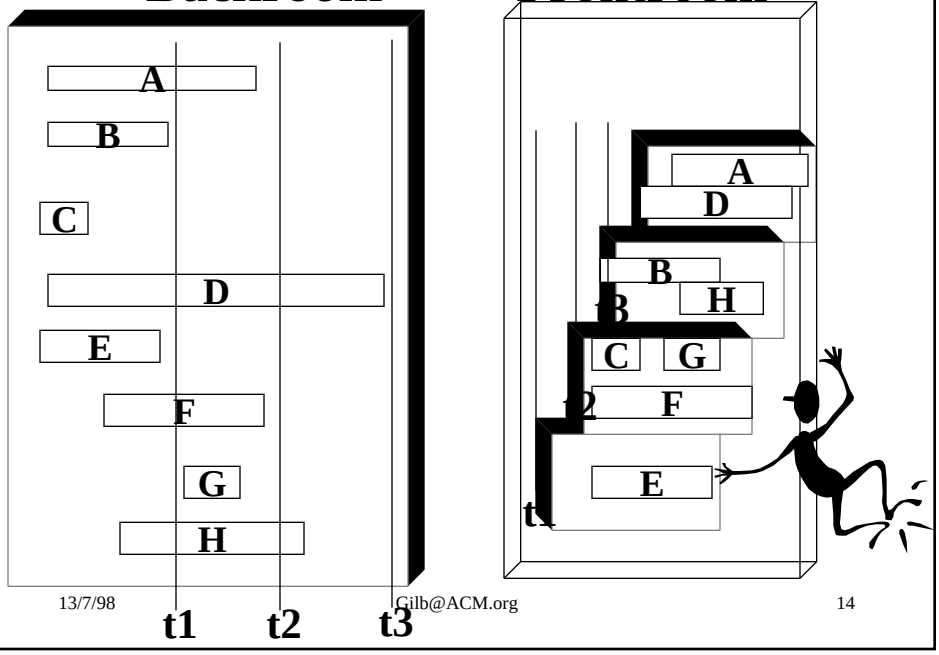
- "Organic Integration"
  - Ericsson, Jack Jarkvik
- Integration testing in 2% increments
- Testing of all aspects of system
  - Performance
  - Reliability
  - Usability
  - Security
- Some kind of internal testing is done before release to Evo user
- Highest Quality levels might be expected each step (Mills Cleanroom)
- Feedback goes to requirements and design improvement
- No artificial test cases
- Heavy user trial involvement
- See Gilb: "Evo: The Evolutionary Project Management Handbook"
  - Www.result-planning.com

"Evo" model



IBM Cleanroom Test Practice

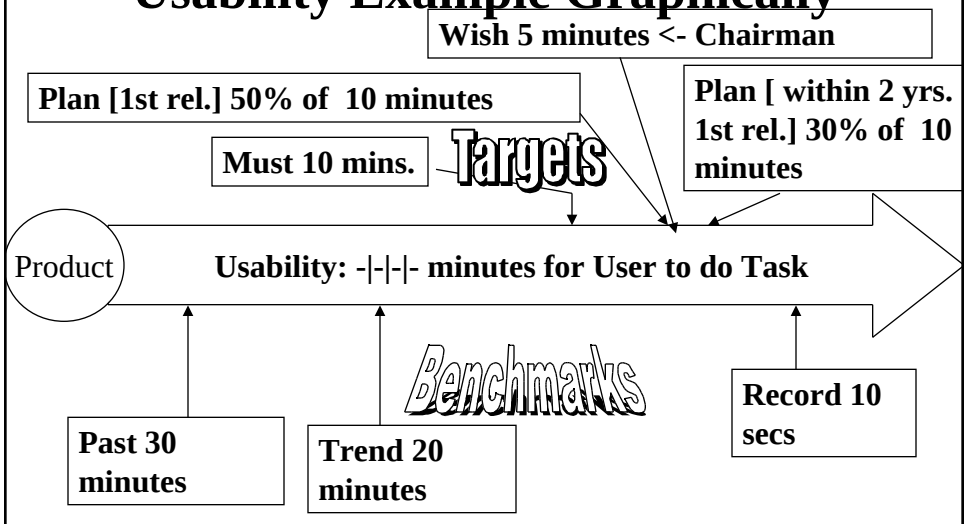
- **"When the development and test of an increment are complete, an estimate to complete the remaining increments is computed.**
  - ***The algorithms used in this computation should reflect the various actual productivity rates experienced in developing and testing previous increments.***
  - ***An alternative plan is prepared and reviewed, as previously described, whenever a cost projection is inconsistent with its cost plan….***
  - ***The design-to-cost practice describes the management control procedures that balance cost, schedule, and functional capability."***
  - ***Robert Quinnan, [IBM Systems Journal Four 1980, page 474]***



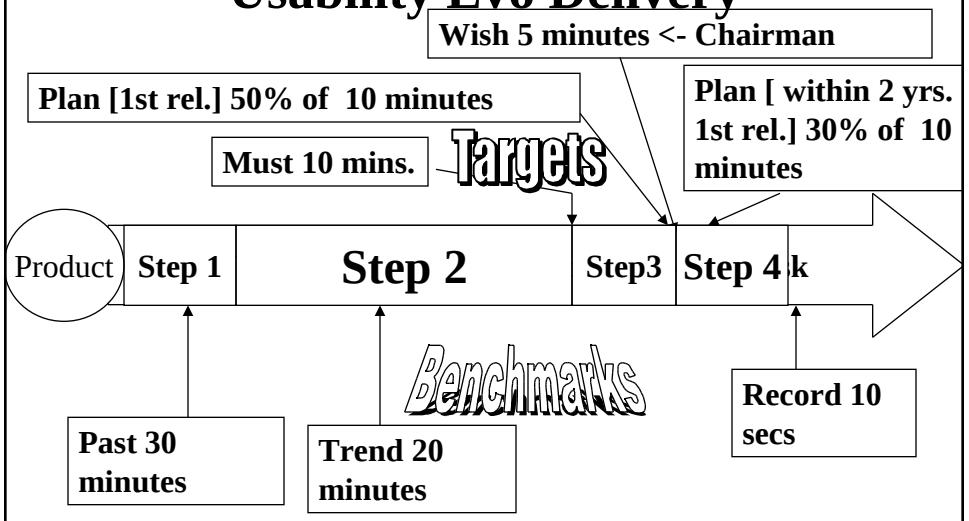
Requirements Refinement (JPL)

- "a key benefit … is its ability to progressively refine requirements
  - and to respond easily to the refinements.
- Refinement is done on the basis of
  - developmental test,
  - training, and
  - operational experience.
- Requirements feedback facilitates working in an environment of change." [SPUCK93]

Usability Example Graphically

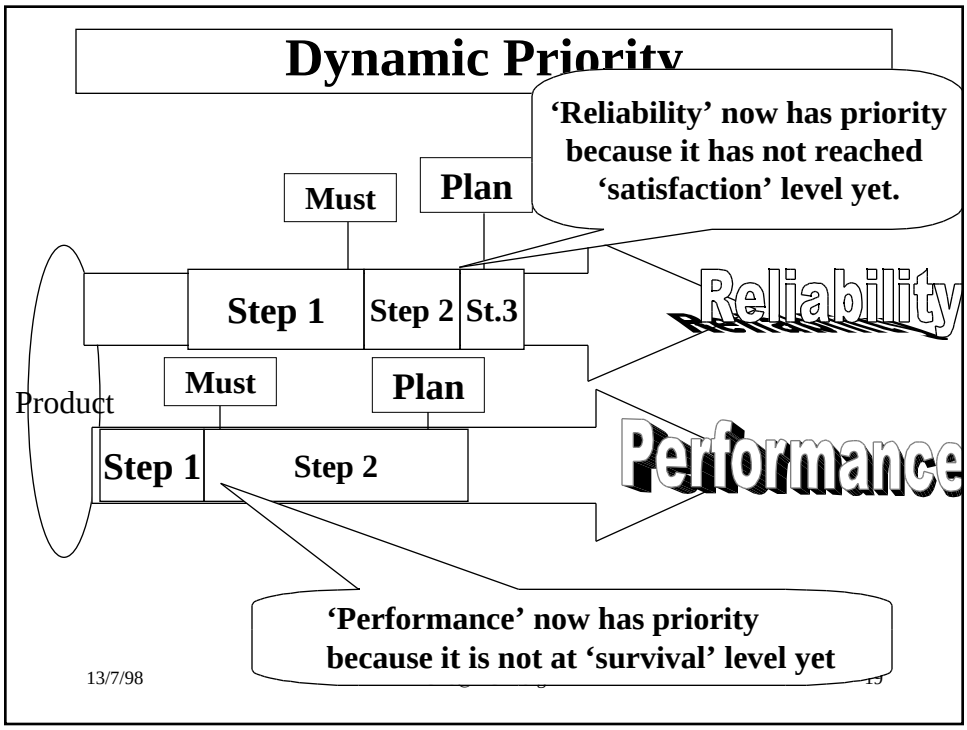


Usability Evo Delivery



An example of a typical one-week Evo cycle at the HP Manufacturing Test Division during a project. [MAY96]

	Development Team	Users
Monday	• System Test and Release Version N • Decide What to Do for Version N+1 • Design Version N+1	
Tuesday	• Develop Code	• Use Version N and Give Feedback
Wednesday	• Develop Code • Meet with users to Discuss Action Taken Regarding Feedback From Version Nû1	• Meet with developers to Discuss Action Taken Regarding Feedback From Version Nû1
Thursday	• Complete Code	
Friday	• Test and Build Version N+1 • Analyze Feedback From Version N and Decide What to Do Next	



Microsoft Pairs up Testers and Developers in Milestone tests

- ***"Development Phase: Feature development in 3 or 4 sequential subprojects that each results in a milestone release.***
- ***----------***
- ***Program managers coordinate evolution of specification.***
- ***Developers design code and debug.***
- ***Testers pair up with developers for continuous testing.***
- ***----------***
  - ***Subproject I First 1/3 of features. Most critical features and shared components.***
  - ***Subproject II Second 1/3 of features.***
  - ***Subproject III Final 1/3 of features. Least critical features."***
- ***Cusumano and Selby, Microsoft Secrets, page 194.***
- ***The point I want to bring out here is the priority sequencing rule at Microsoft. A milestone is a 6-10 week segment.***

HP on Releasing after Evo cycles

- **"With an Evo approach, the team has greater flexibility as the market window approaches.**
- **Two attributes of Evo contribute to this flexibility.**
  - **First, the sequencing of functionality during the implementation phase is such that "must have" features are completed as early as possible, while the "high want" features are delayed until the later Evo cycles.**
  - **Second, since each cycle of the implementation phase is expected to generate a 'complete' release, much of the integration testing has already been completed.**
  - **Any of the last several Evo cycles can become release candidates after a final round of integration and system test.**
  - **When an earlier-than-planned release is needed, the last one or two Evo cycles can be skipped as long as a viable product already exists.**
  - **If a limited number of key features are still needed, an additional Evo cycle or two can be defined and implemented**…"
  - [COTTON96, HP Journal August 1996]

Daily Build Process at Microsoft

- ***.Check Out [copies of code from master version]***
- ***.Implement Feature.***
- ***.Build Private Release.***
- ***.Test Private Release.***
- ***.Synch Code Changes [use compare tool to make sure no changes since checkout].***
- ***.Merge Code Changes.***
- ***.Build Private Release [merged with other developers' changes]***
- ***.Test Private Release***
- ***.Execute Quick Test.***
- ***.Check In.***
- ***.Generate Daily Build.***
- ***.Execute automated tests***
- ***.Make Build Available to All Project Personnel***

(including program managers, developers, testers, and user education staff for their use and evaluation)"

Headlines only, from *Cusumano and Selby, Microsoft Secrets*, page 264-7

HP Evo Experiences

- *"...and the team attributed several positive results to having used the Evo method for the majority of the project.*
- *First, Evo contributed to creating better teamwork with users and more time to think of alternative solutions.*
- *Second, the project still had significantly fewer critical and serious defects during system testing.*
- *Third, the team was surprised to see an increase in productivity (measured in [non-commentary lines of code] per engineer-month).*
  - *The project manager attributes this higher productivity primarily to increased focus on project goals."*
- *[MAY96, Elaine May, HP Journal Aug 1996, on hp web]*

A4. Defect Detection Inspections

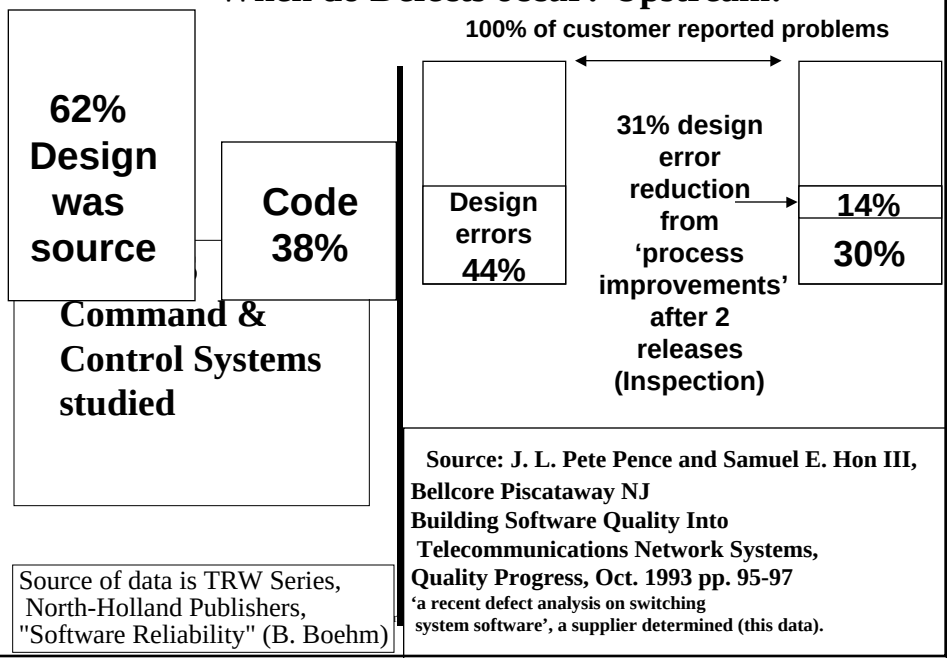
- **95% of injected defects can be removed before first test (IBM, Sema)**
- **44%->62% of all code defects which escape test to field, are due to requirements/design specification defects (TRW, Bellcore).**
- **40% 0f all development effort is retesting due to bugs.**
  - **This can be avoided using Inspections (Raytheon 95)**

Costs of Non-conformance Items

- **Re-reviews**
- **Re-tests**
- **Fixing Defects (code, documentation)**
- **Reworking any document.**
- **Engineering Changes**
- **Lab Equipment Costs of Retests**

- **Updating Source Code**
- **Patches to Internal Code**
- **Patches to Delivered Code**
- **External Failures**
- **from Crosby's Model according to Raytheon95 Fig. 7**

When do Defects occur? Upstream!



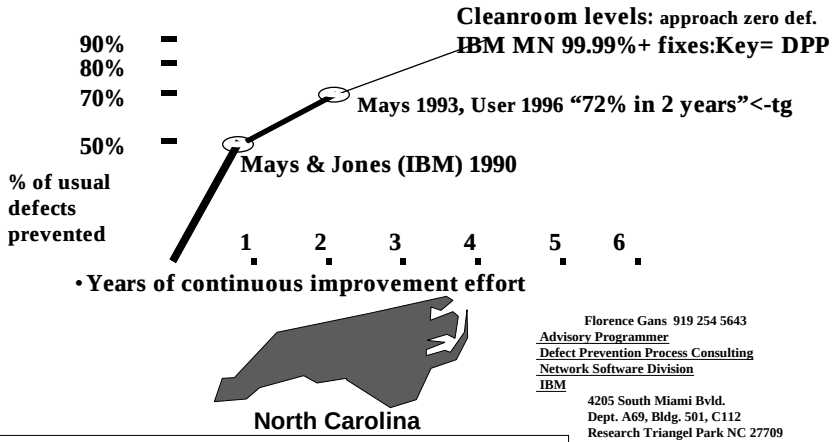
Source of data is TRW Series, North-Holland Publishers, "Software Reliability" (B. Boehm)

Source: J. L. Pete Pence and Samuel E. Hon III, Bellcore Piscataway NJ
Building Software Quality Into Telecommunications Network Systems, Quality Progress, Oct. 1993 pp. 95-97
'a recent defect analysis on switching system software', a supplier determined (this data).

A5. Defect Prevention Process

- DPP (= CMM 5), Mays and Jones IBM
- Can be applied to any 'test' process
  - So as to make it more effective
- Can be applied to any software process
  - So as to reduce test bugs and difficulties
- DPP feeds off of all defect data
  - From all tests
  - From inspections
  - From field reports
- Basic idea: systematic analysis of 'root' cause
  - And changing the work process to avoid the cause

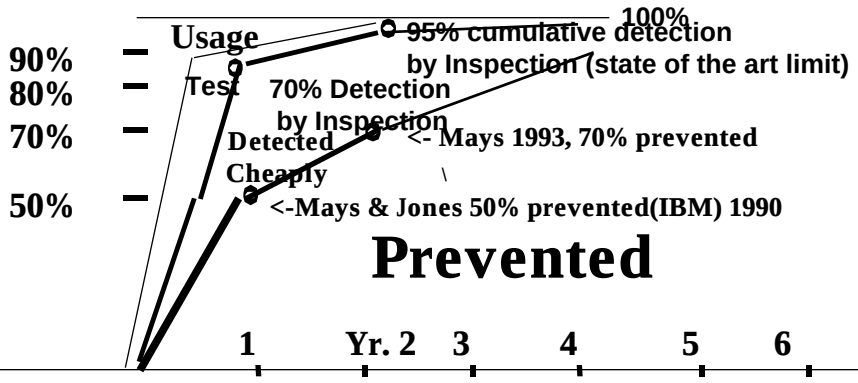
Defect Prevention Experiences: Most defects can be prevented from getting in there *at all*



Florence Gans 919 254 5643
Advisory Programmer
Defect Prevention Process Consulting
Network Software Division
IBM
4205 South Miami Bvld.
Dept. A69, Bldg. 501, C112
Research Triangel Park NC 27709
flogans@us.ibm.com (1998 new one)

IBM Research Triangle Park Networking Laboratory

Prevention + Pre-test Detection
is the most effective and efficient bug avoidance method



- **Prevention data based on state of the art prevention experiences (IBM RTP), Others (Space Shuttle IBM SJ 1-95) 95%+ (99.99% in Fixes)**
- **Cumulative Inspection detection data based on state of the art Inspection (in an environment where prevention is also being used, IBM MN, Sema UK, IBM UK)**

A Supplement to testing

- **Can be added to current test process**
- **And would make them more effective**
- **Or less costly**

Some supplements to Conventional Testing

- s1.Formal Test Entry
- s2. Formal Test Exit
- s3. Inspection of Test plans and test cases
- s4. Quality Testing
- s5. Evolutionary field trial deliveries
- s6. Design for Testability
- s7. Inspection of Changes
- s8. Robust design
- s9. Motivation
- s10. Rework cost reduction
- s11. Continuous Improvement
- s12. Managing Test Process Qualities
- s13. Design for Quality
- s14. Defined Rules for Test Planning
- s15. Impact Estimation

s1.Formal Test Entry

- Entry Conditions
  - Minimum quality levels of documentation from which test plans and cases are derived
    - Example: 'Maximum 1.0 Major defects/page remaining'
  - More generally: any technical document used has exited formally from its own inspection process.
    - This implies that the documents are trustworthy and can economically be used to generate tests

s2. Formal Test Exit

- Can be used to prevent 'false' ends to test planning or to test process
- For test planning,
  - Use 'Inspection' (possibly a sample)
  - Based on Major defects found and fixed:
    - Calculate remaining Major defect/Page
    - Set a threshold level for exit, ex. "Max 1.0/Page'
- For testing processes
  - Set numeric levels of critical parameters
    - Requirements, Coverage, Volume, Bugs, Corrections
    - Calculate critical levels of combinations of these

98.7% Detection Effectiveness: Bull

- Project D with 18,000 LOC, data access service module, 112 modified modules.
- 298 Major defect found by inspections
- no bugs found in first six modified modules
- based on this sample they skipped unit test
- Only 4 defects found later (one year test & use)
- 298/302 = 98.7% inspection effectiveness.
- Code insp. pretest 80%, post test 70% (Proj C)
- Source Ed Weller IEEE Software Sep.1993

s3. Inspection of Test plans and test cases

- IBM Experience Unit Test Inspections
  - 82% of human effort saved
  - 93% computer time saved
  - As a result of 'forcing better test organization'MF
- Inspection based on defined best practice rules for planning tests
  - Teaches the best rules
  - Reinforces through feedback when not followed
  - Is a basis for continuous improvement of the Rules

s4. 'Quality' Testing

- Test a set of qualitative aspects of your system
  - Not just functionality and performance
  - Security, Usability
  - Installability, Maintainability
  - Mean time to Failure (Reliability).
- Validate the resultant qualities of the system 'functionality', not just that the functions intended to give quality are present.
- You should have agreed requirements definitions of these things to base your tests on.

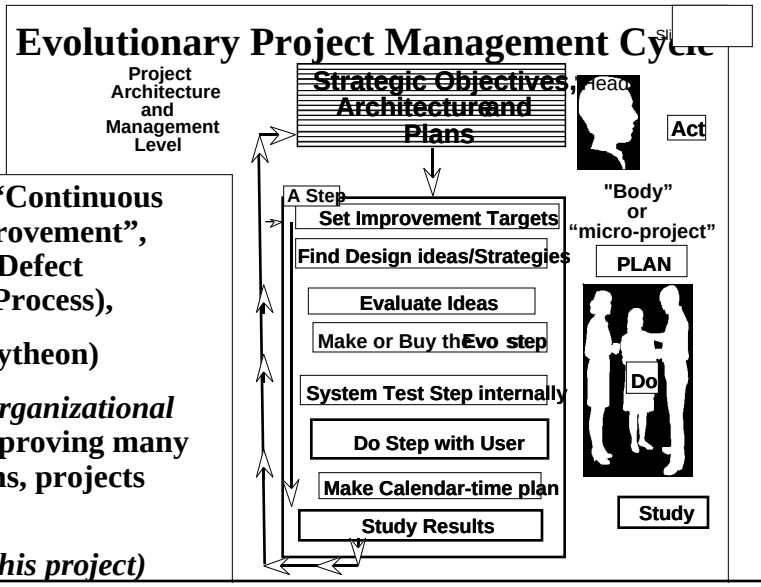
PORTABILITY

- PORTABILITY:
- **GIST: portability of all test scripts and tools to defined customer environment <-Leon Osterweil's idea Univ Amherst**
- **SCALE: hours per 100 test cases of conversion to [defined** platforms**] using [defined** methods **or tools and human efforts] using [defined human** skill **levels]**
- **METER: by selected samples of past and current test porting efforts.**
- **PLAN [to UNIX, SRA Tools, NOVICE TRAINEE CUSTOMER TESTERS]** 2 **hours "per 100 test cases converted"**

s5. Evolutionary field trial deliveries

- **Evolutionary delivery steps, frequently, early**
- **Confirms testing result, with real users**
- **Done as system is built, not at end**
- **Can generate realistic test scripts for regression tests**
- **Can test things like performance and usability which are difficult to test in lab**
- **Can point out weaknesses in testing plans and cases**

Evo Cycles



Evo allows "Continuous *Project* Improvement", while DPP (Defect Prevention Process),

(used by Raytheon)

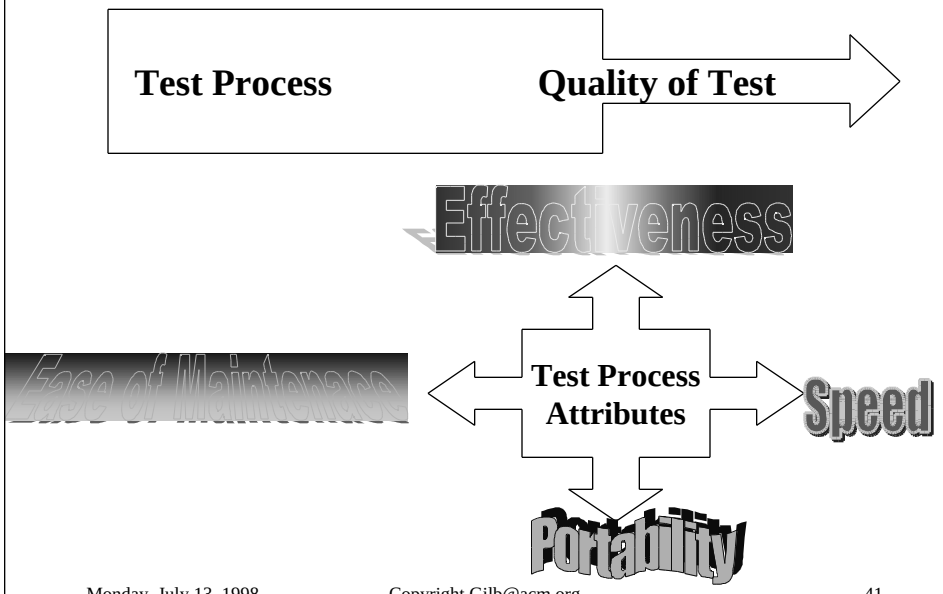
focuses on *organizational learning*, improving many people, teams, projects later;

(maybe not this project)

s6. Design for Testability

- **Design both your product and your processes for ease of testing**
- **Begin by quantifying your 'ease of test' objectives.**
- **Design to meet the objectives.**
- **Test (measure) that you are meeting them**
- **Relate objectives to costs of not meeting them**
  - **To ensure motivation and funding to work this aspect**
  - **Calculate return on investment to motivate continuance of the effort**

Test dimensions



1st Area of application: Test Goodness Measures
Quality of a Test Process, a brainstormed list by participants 1996

- ***Effectiveness*** **(% of existing defects found)**
- ***Efficiency*** **(bugs per work hour?)**
- ***Time*** **to release** ***to market***
- ***Reliability*** **of product (mean time to fail for specified users)**
- ***Maintainability*** **(mean time to fix a defect) of test script**
- **automatic testability**
- ***Remaining defects*** **afterwards for system lifetime**
- ***Simplicity*** **of executing the tests by defined test people.**
- ***Accuracy*** **of testing intended things**
- ***Portability*** **of test materials to other environments, platforms**

An example of definition: Test Effectiveness Measures

- **TEST-EFFECTIVENESS "for a defined process"**
- **GIST: what portion of defined defect types are we finding using our test processes?**
- **SCALE: % of [DEFINED DEFECTS] identified by DEFINED TEST PROCESSES within DEFINED EFFORT using DEFINED TOOLS**
- **METER: <sampling of test efforts by QA>**
- **PAST [LOGICAL BUGS, BRANCH COVERAGE TESTING, 80% level effort, SRA Tools] 50% + or - 30%?? <- Tom Gilb wild illustrative guess to provoke better information.**
- **PLAN [ANY DESIGN DEFECT, {Inspection and Our Tests}, 1 work hour per 100 LOC, NO SPECIAL TOOLS] 30% ?? <- swag tg**
- **DEFECTS:DEFINED: any difference from formal requirements.**
- **DESIGN DEFECT: DEFINED : any difference from specified design.**

s7. Inspection of Changes

- **IBM found Inspection of changes 2X more profitable than Inspection of new code (MF 1984)**
- **IBM (Kan MN) and Space Shuttle have achieved about 99.99% correct corrections by Inspecting the fixes before testing them.**
- **You need to base the Inspections on a state of the art Standard**
  - **(Rules, Checklists, Optimum Checking Rates).**
  - **Must use Input Sources (changes, old system)**
  - **A solid Inspection Process (See Optimizing Inspection, DoD Crosstalk March 1998), not the commonly misused Inspection process (bad checking rates, bad Exit Entry control, inadequate Rules specification)**

s8. Robust design

- Consider designing your system to tolerate bugs! (but 'test people' don't get to do this!)
- Distinct Software or N-version programming is one possibility
- Larger amounts of logic checking reasonableness
- Better Object encapsulation
- Built-in testing mechanisms (perhaps you can!)
- Database diagnosis, maybe in real time

s9. Motivation

- "Motivation is Everything!"
- Have you looked at your organization's motivation
  - To test well enough
  - To test early and frequently
  - To do continuous improvement on testing?
  - To avoid injecting upstream defects into test
    - Code, requirements, design specs, user manuals
- How are people paid and rewarded?
- Is there a team developers and testers?
  - Or a wall, us and them?

HP Motivation & Evo test

- "Some of the gains in productivity seen by project teams using Evo have been attributed to higher engineer motivation.
- The long implementation phase of the waterfall life cycle is often characterized by large variations in engineer motivation.
- It is difficult for engineers to maintain peak productivity when it may be months before they can integrate their work with that of others to see real results.
- Engineer motivation can take an even greater hit when the tyranny of the release date prohibits all but the most trivial responses to customer feedback received during the final stages of system test."
- COTTON96, Todd Cotton, HP Journal August 96
  - Available via web.

s10. Rework cost reduction

- Most testing effort is in fact unnecessary rework (regression after fixing)
- Most of the rework can be systematically reduced (10x) by hard work in improving the processes which generate it
  - Improve requirements processes
  - Improve Inspections upstream
  - Improve reuse of code and other systems
- Maybe one day testing can do what it is supposed to do properly
  - Instead of cleaning up other peoples dirty laundry!

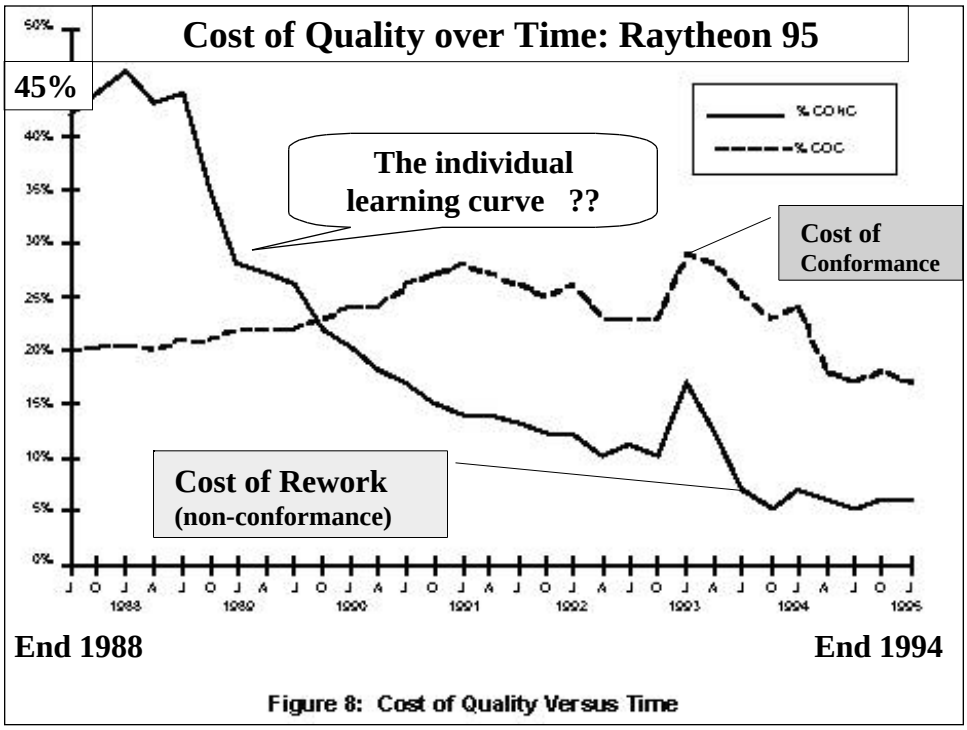


Figure 8: Cost of Quality Versus Time

s11. Continuous Improvement

- Are you part of a serious continuous improvement effort
  - One that is measured in terms of effects
  - And return on investment
- Is your effort specifically directed towards testing?
- Is it also directed towards the things that impact testing
  - Upstream defects
  - Upstream documentation

IBM MN & NC DP Experience

- 2162 DPP Actions implemented
  - between Dec. 91 and May 1993 (30 months)<-Kan
- RTP about 182 per year for 200 people.<-Mays 1995
  - 1822 suggested ten years (85-94)
  - 175 test related
- RTP 227 person org<- Mays slides
  - 130 actions (@ 0.5 workyears
  - 34 causal analysis meetings @ 0.2 workyears
  - 19 action team meetings @ 0.1workyears
  - Kickoff meeting @ 0.1 workyears
  - TOTAL costs 1% of org. resources
- ROI DPP 10:1 to 13:1, internal 2:1 to 3:1
- Defect Rates at all stages 50% lower with DPP

s12. Managing Test Process Qualities

- Assuming you have defined your desired 'test process qualities' ,
  - Does someone 'own' the test processes
  - Is there a *budget* for test process improvement effort?
  - Is there *motivation* to make it happen, *this* year?
- Is test part of the larger effort to improve systems
  - Or is it focussed on development ex test
  - Is it focussed on the total lifecycle costs?

s13. Design for Quality

- Is there a design process which has quantified quality targets ?
- Does your organization consciously design various quality aspects into product and process?
- Do you have quality estimation ability during the design phase (like 'Impact Estimation')?
- Does anybody actually test that the design-in qualities are present?
  - Do they test these qualities evolutionarily?

s14. Defined Rules for Test Planning

- Is there a strong set of Generic technical documentation rules which apply to test planning as well
  - On subjects like tagging, cross referencing, clarity
- Is there a strong set of specific rules in addition for specific types of test planning
  - Test strategies/plans
  - Test scripts
  - Planning maintenance change tests
- If not, how do people learn best practices rapidly in a large organization?

Generic (Engineering Specification) Rules ID:S.Rules.GEN 1/4

- **GEN.0 Generic Rules: Defined:**
  - **Generic Rules are engineering specification rules applying to all engineering documents as required best practices. They are separated from related Specific Rules so as to avoid repeating them, and to permit learning them well.**
- **GEN.1 (Consistency)**
  - **Specifications must be consistent with all other related specifications in the same document, and in all related documents {Sources, Kin and Children}, otherwise potential defects are to be suspected.**
- **GEN.2 (Unambiguous)**
  - **All specifications must be unambiguous to the Intended Readership, as practiced or implied, for that document.**
- **GEN.3 (Notes)**
  - **All manner of comment, notes, suggestion or ideas which are not themselves the actual engineering specification, but merely background, shall be clearly distinguished as such by suitable devices.**
  - ***Suggested Devices: italics, "double quotes", Note:…, Comment:…., use of footnotes, use of separate commentary pages.***
  - ***In particular defects in a note should never result in a Major Defect.***

Generic Rules Page 2

- **GEN.4 (Brevity)**
  - **All specification shall be as brief as possible, in order to successfully support their purpose, for the defined Intended Readership.**
- **GEN.5 (Clarity of Purpose)**
  - **All specification shall result in clarity to the Intended Reader regarding it's purpose or intent.**
  - **Note: it is not enough that the statements are unambiguous. They must contain clarity of purpose: why is this here?**
- **GEN.6 (Elementary)**
  - **Specification statements shall be broken up into their most elementary form.**
  - ***Note: this is so that they each can be unambiguously cross-referenced externally.***
  - ***Note: an elementary specification can be refered to by its tag (or global tag), and within that, any unique combination of Parameter (for example Meter, Past) and/or Qualified (for example "[UK, 1999]") may be used to refer to or distinguish an elementary idea.***
- **GEN.7 (Unique)**
  - **Specifications shall only have a single instance in the entire project documentation.**

Generic Rules Page 3 of 4

- **GEN.8 (ID Tags)**
  - **All specifications statements must have some unique and unambiguous cross-reference capability, which is stable (independent of sequence and changes).**
  - ***Statements may be directly tagged.***
  - ***Statements may be referenced by a hierarchy of tags***
  - ***Statements may be referenced in detail using Parameter names (like Meter, Plan) and Qualified (like [Euro, 2001])***
  - ***References may have defined synonyms for convenience.***
- **GEN.9 (Reuse)**
  - **Tagged Ideas shall be 'reused' by cross-reference to their unique tag, or their tag and version information.**
- **GEN.10 (Source)**
  - **Specifications shall contain detailed (paragraph, unique tag level) about their exact and detailed sources.**
  - ***Note use: 'Spec <- Source' format or 'Source:……'***
- **GEN.11 (Hierarchy)**
  - **Specifications shall be logically organised into a useful hierarchical structure.**
  - **The tagging system shall reflect that structure (example A.B.C)**
  - **Note: this may be partly done through templates.**

Generic Rules page 4 of 4

- **GEN.12 (Auditability)**
  - **All changes to specifications shall be done in such a way as to permit us to know who changed things, what the previous version was, perhaps even justification for the change.**
- **GEN.13 (Risk)**
  - **The specification Author must clearly indicate any information which is uncertain or poses any risk to the project whatsoever, using a variety of available devices such as {<fuzzy brackets>, ??, ?, ranges (60->70, 60±10%), suitable comments or notes}.**
- **GEN.14 (Template)**
  - **The relevant electronic template shall be used, for specific headings and guidance under each heading.**
- **GEN.15 (Model)**
  - **A 'best practice' model shall be defined and available. It shall be used to help interpret the intent of these rules in practice.**

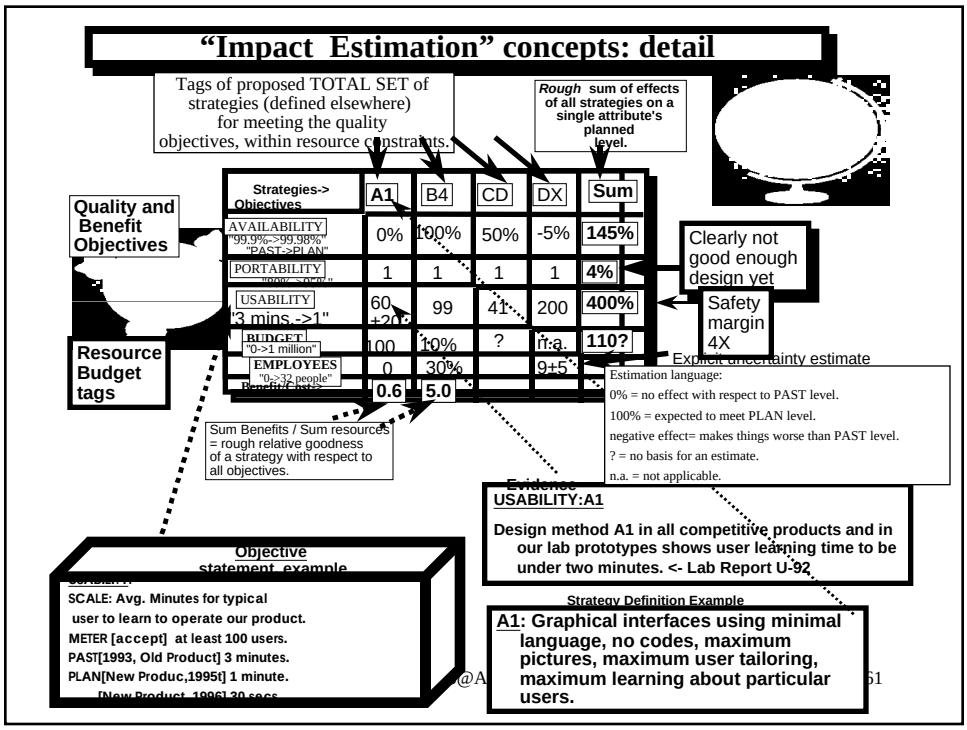
s15. Impact Estimation

- Impact estimation tables can be used to systematically evaluate all test processes, tools, methods and alternatives
- The basis for evaluation is your tailored objectives for test, test processes, current project objectives, component objectives
- Test 'means' are evaluated in relation to these objectives, and with regard to other test 'means' (strategies) being considered.
- It forces people to think about what they know, or do not. It is more objective.

"Impact Estimation" concepts: full table

Tags of proposed TOTAL SET of strategies (defined elsewhere) to meet objectives, within resource constraints.

Strategies-> Objectives	A1	B4	CD	DX	Sum
AVAILABILITY "99.9%->99.98%" "PAST->PLAN"	0%	100%	50%	-5%	**145%**
PORTABILITY "80%->95%"	1	1	1	1	**4%**
USABILITY "3 mins.->1"	60 ±20	99	41	200	**400%**
BUDGET "0->1 million"	100	10%	?	n.a.	**110?**
EMPLOYEES "0->32 people"	0	30%		9±5	
Benefit/Cost->	**0.6**	**5.0**			



So what?

- I hope this gives you a rich set of ideas to consider for improving your software development process, especially the testing process.
- Maybe one of them is worth working on from next week? (well you can at least think about it!)
- The others can be on your evolutionary list of things to do.
- More detail in the references! :)> Tom

Some References

- **RPL Site: www.Result-Planning.com**
  - **Gilb web site with many papers, slides, book manuscripts to supplement this talk.**
- **Software Inspection: by Gilb and Graham**
  - **Addison-Wesley Longman 1993**
- **Raytheon95: SEI Website report**
  - **Ftp.sei.cmu.edu/pub/documents/95.reports/pdf/tr017.95.pdf**
- **PoSEM: Gilb: Principles of Software Engineering Management, Addison-Wesley Longman, 1988**